RISC V Processor

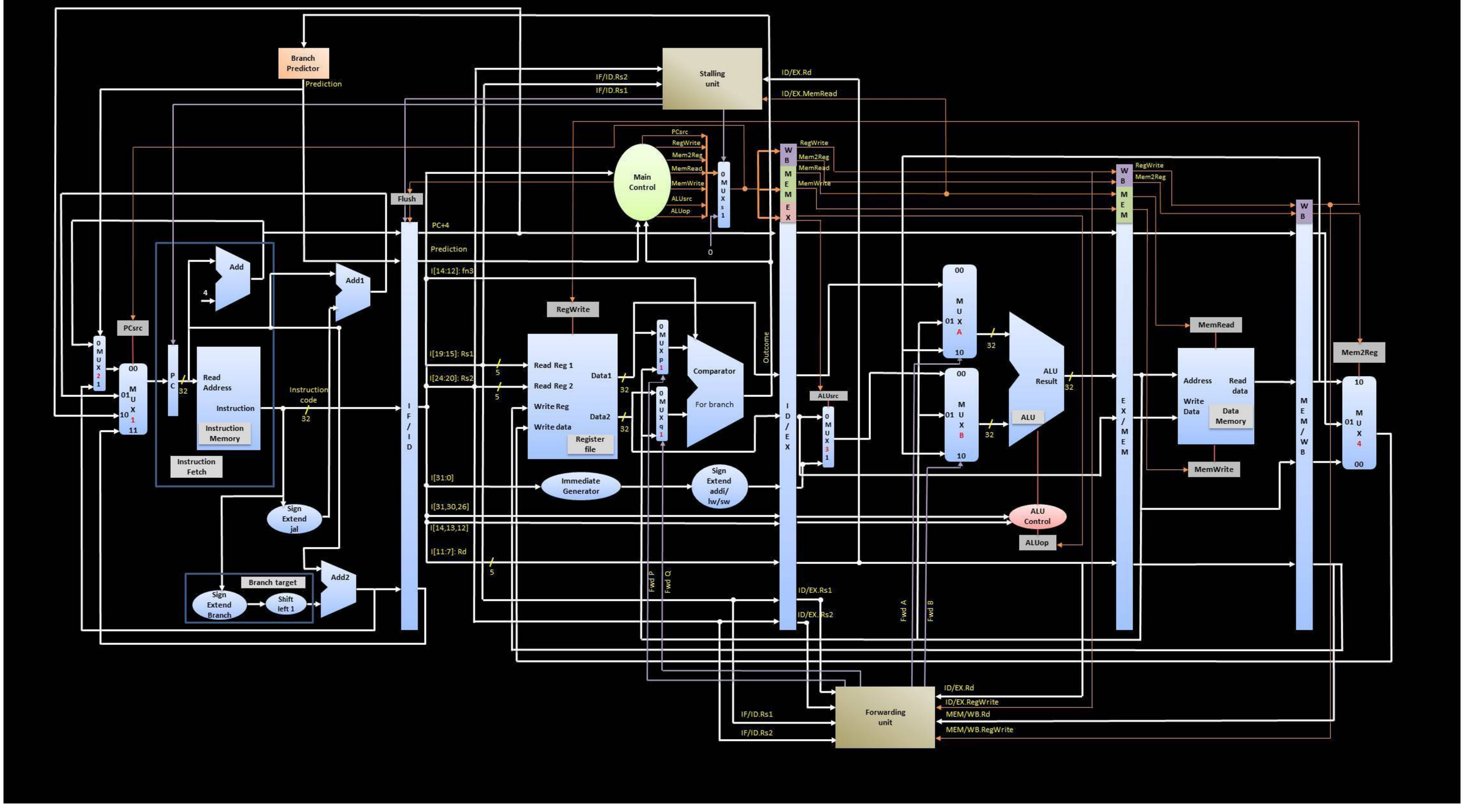
Design details

- Anirudh Iruvanti

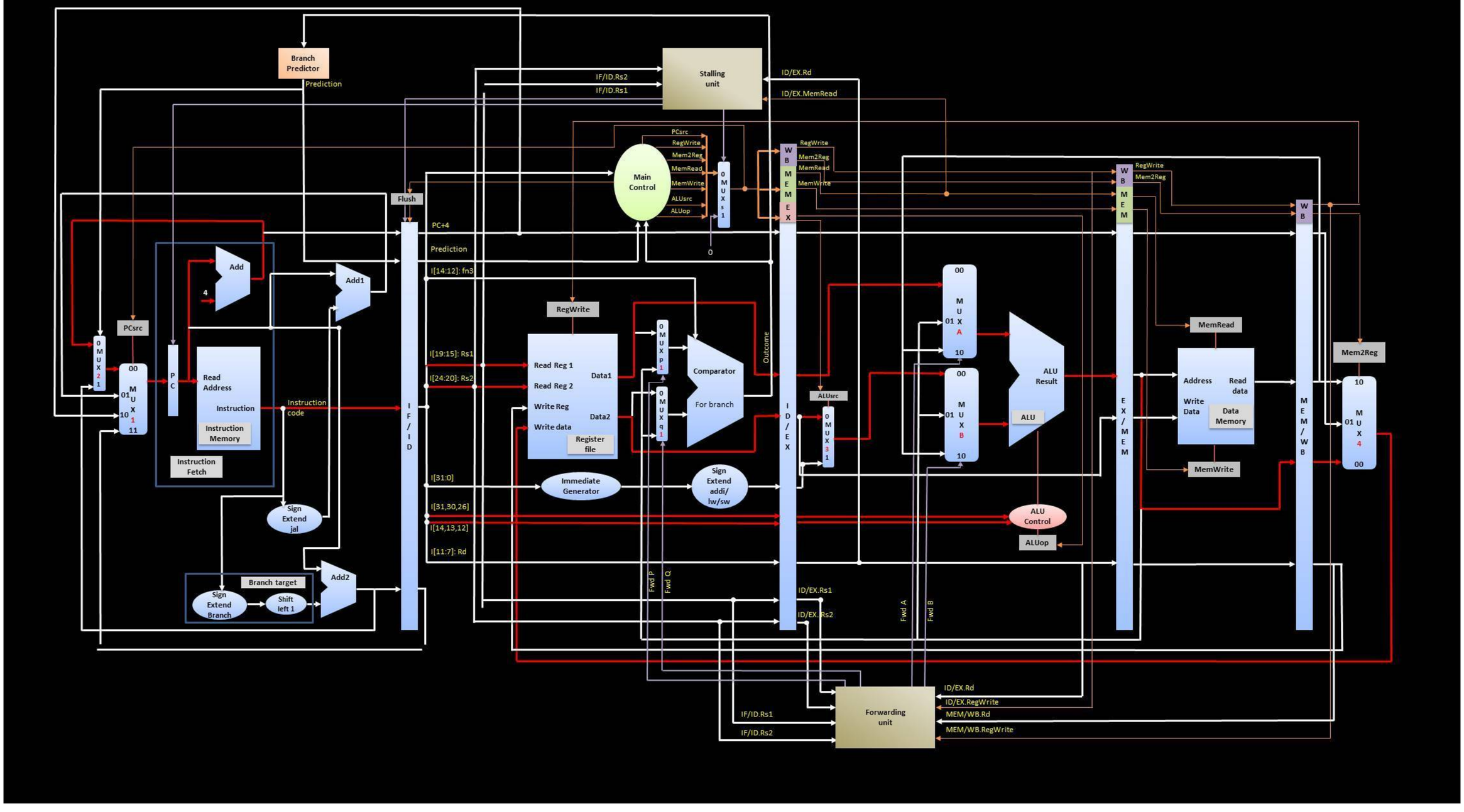
Instructions supported (15)

Instruction type	Instruction
R-type	ADD, SUB, MUL, AND, OR, XOR, SLL, SRL
I-type	LW, ADDI
S-type	SW
J-type	JAL
B-type	BEQ, BGE, BLT

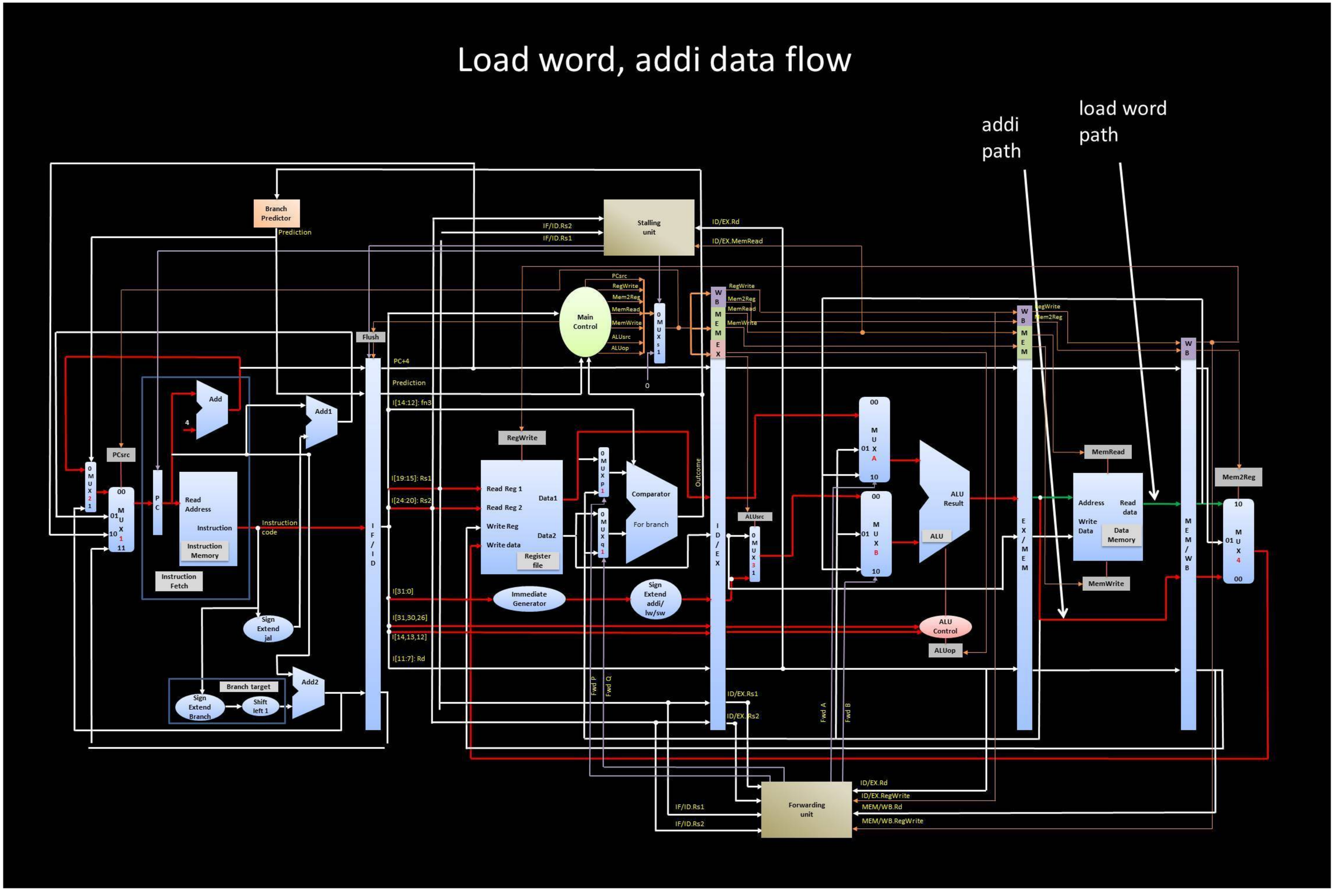
RISC V (32 bit, 5 staged, Pipelined) Processor



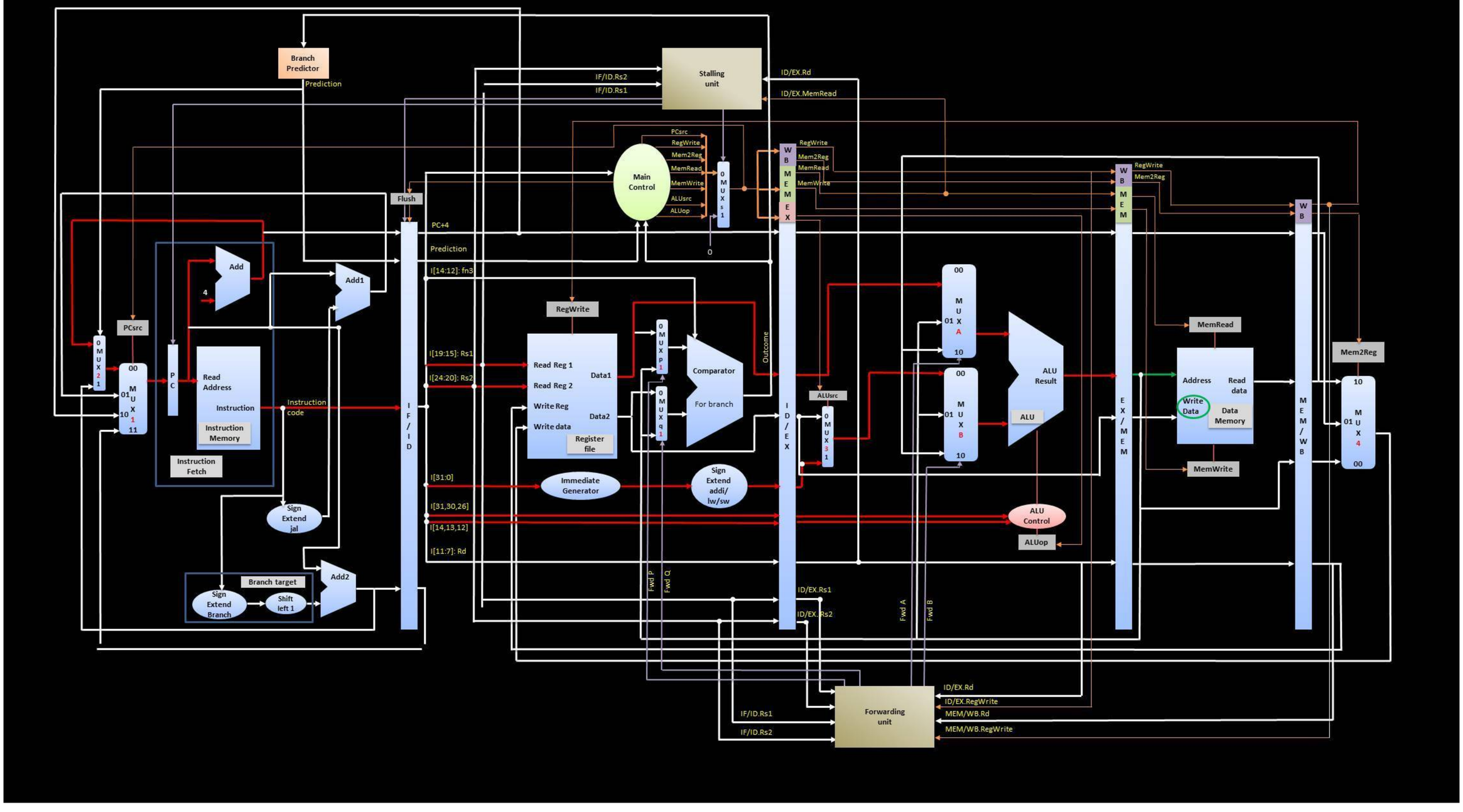
R Type data flow



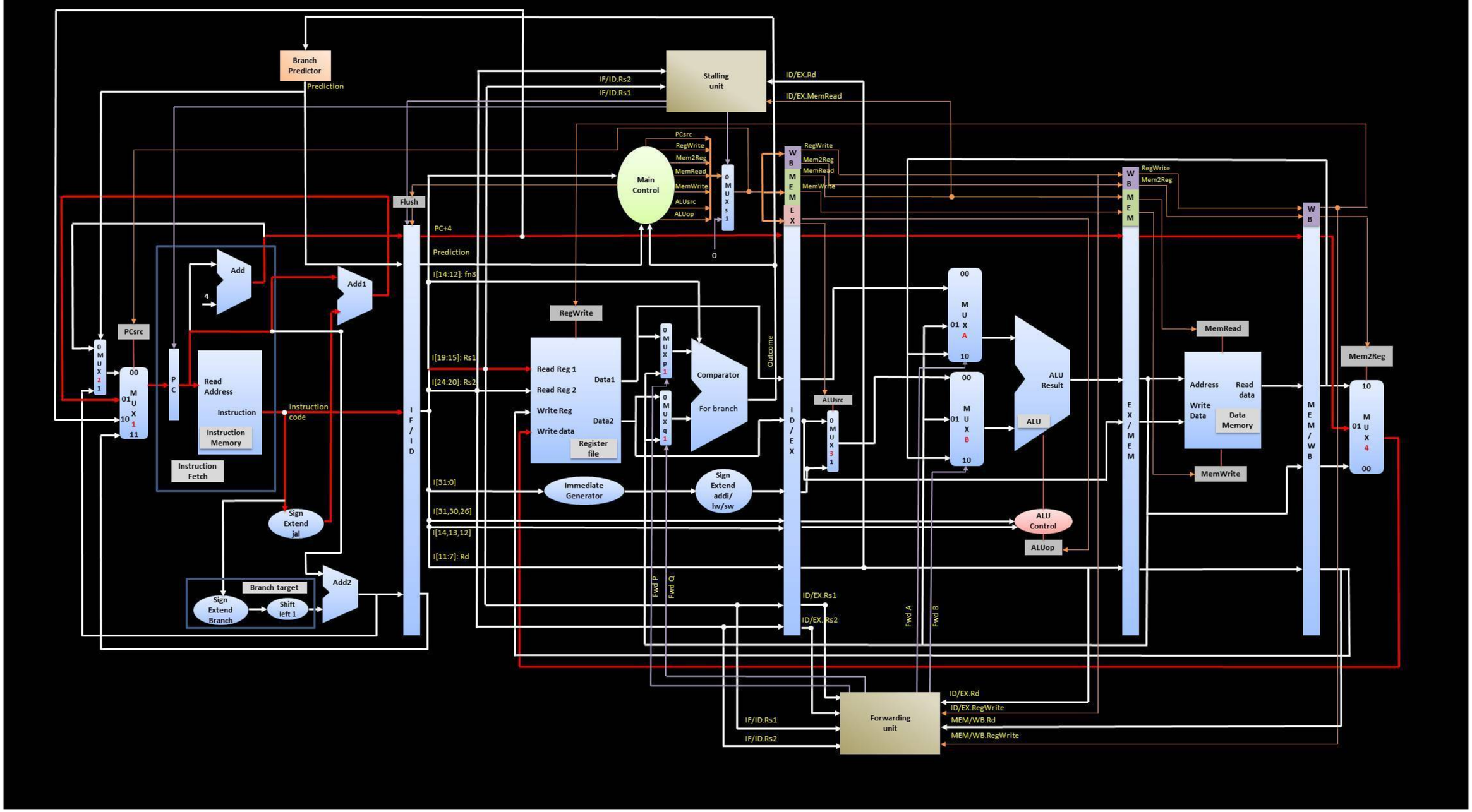
Load word, addi data flow



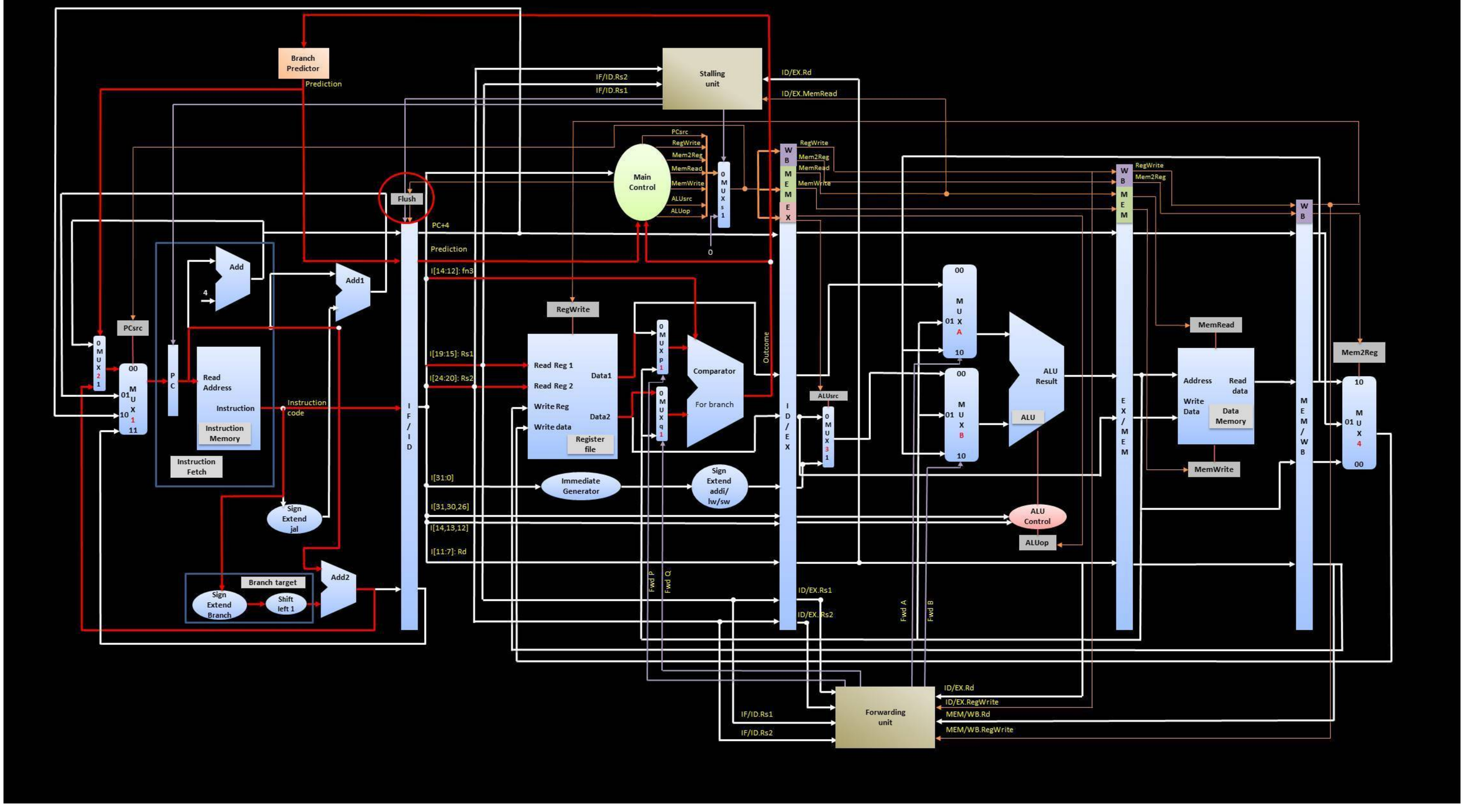
Store word data flow



jal data flow



Branch data flow



ALU control

	ALUop		fn7							fn3			ALUctrl
Func	ALUop1	ALUop0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	ALUctrl
addi/lw/sw	0	0	x	x	x	x	x	x	x	x	x	x	0010
jal	1	0	x	x	x	x	x	x	x	x	x	x	`0010
add	1	0	0	0	0	0	0	0	0	0	0	0	0010
sub	1	0	0	1	0	0	0	0	0	0	0	0	0110
AND	1	0	0	0	0	0	0	0	0	1	1	1	0000
OR	1	0	0	0	0	0	0	0	0	1	1	0	0001
mul	1	0	0	0	0	0	0	0	1	0	0	0	0011
sll	1	0	0	0	0	0	0	0	0	0	0	1	0100
srl	1	0	0	0	0	0	0	0	0	1	0	1	0101

Main control

Instr	op6	op5	op4	op3	op2	op1	op0	PCsrc	ALUsrc	Wr_dt_sel	RegWr	MemRd	MemWr	ALUop1	ALUop0
R-type	0	1	1	0	0	1	1	`00	0	`00	1	0	0	1	0
lw	0	0	0	0	0	1	1	`00	1	10	1	1	0	0	0
sw	0	1	0	0	0	1	1	`00	1	xx	0	0	1	0	0
jal	1	1	0	1	1	1	1	`01	1	`01	1	0	0	1	0
addi	0	0	1	0	0	1	1	`00	1	`00	1	0	0	0	0
branch	1	1	0	0	0	1	1	`10	x	xx	0	x	x	x	x

R-type testing

==========

add x4,x3,x2

sub x5,x3,x2

AND x6,x3,x2

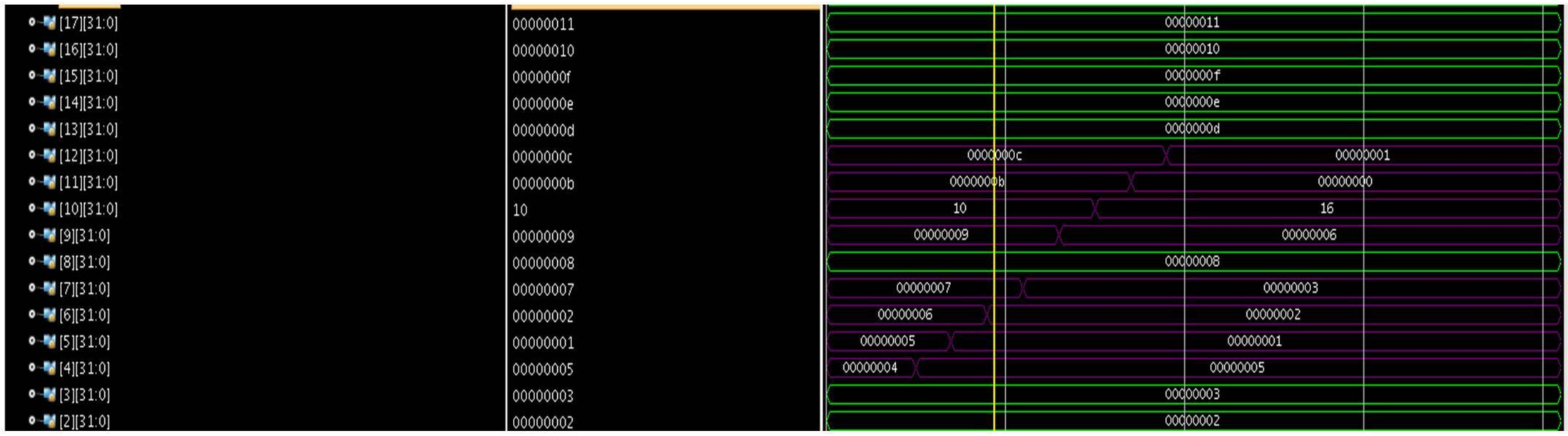
OR x7,x3,x2

mul x4,x3,x2

sll x10,x3,x2

srl x11,x3,x2

XOR x12,x3,x2



Load word

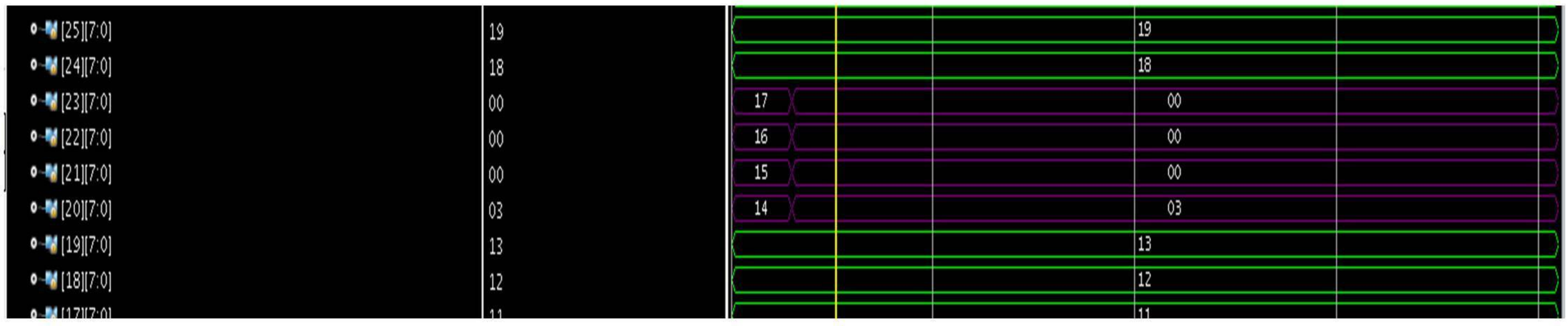
=========

lw x8,0(x2)

Store word

==========

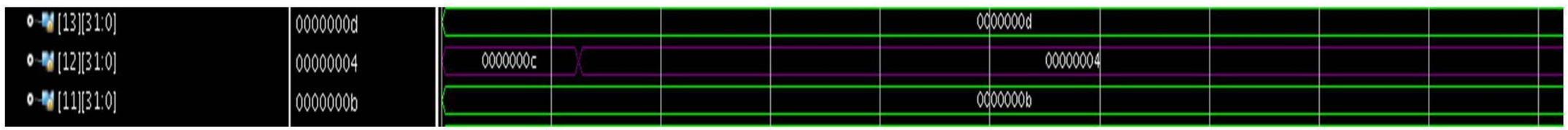
sw x3,0(x20)



Add immediate

==========

addi x12,x3,1



jal test

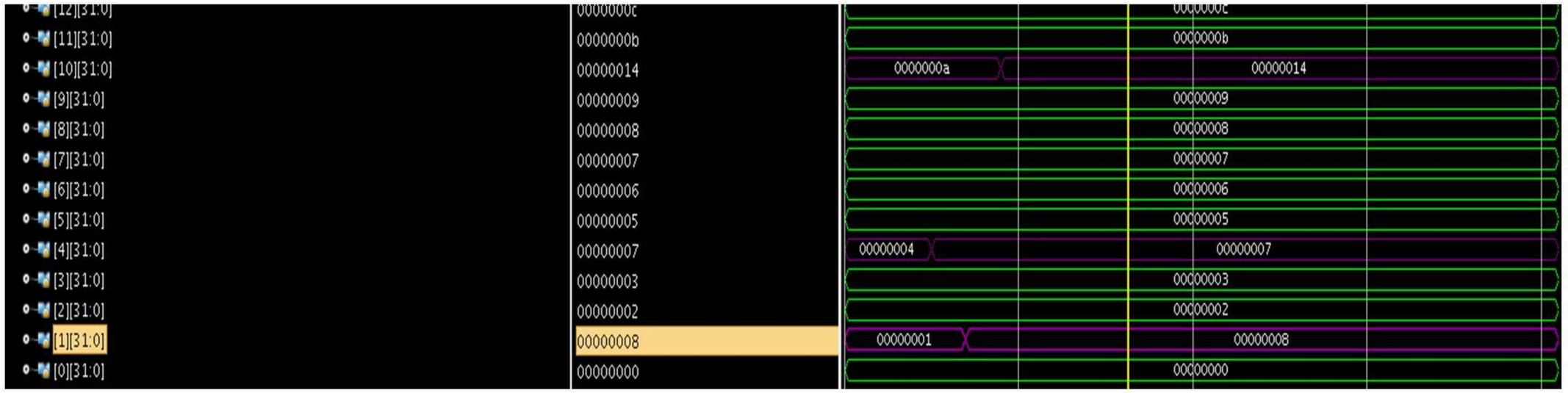
=======

[00] add x4,x5,x2

[04] jal x1,8

[08] add x7,x6,x3

[0C] add x10,x9,x11

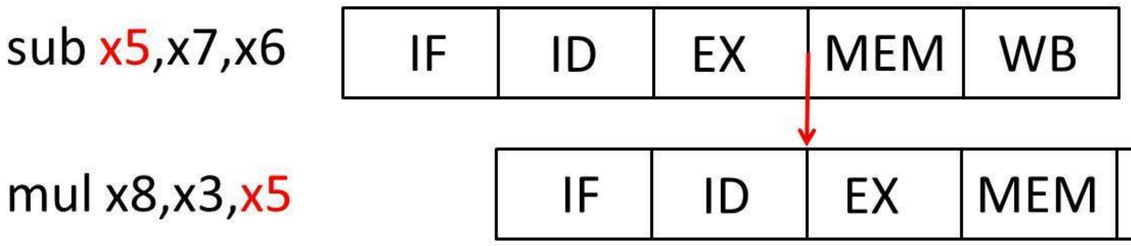


forwarding test 1

==========

sub x5,x7,x6

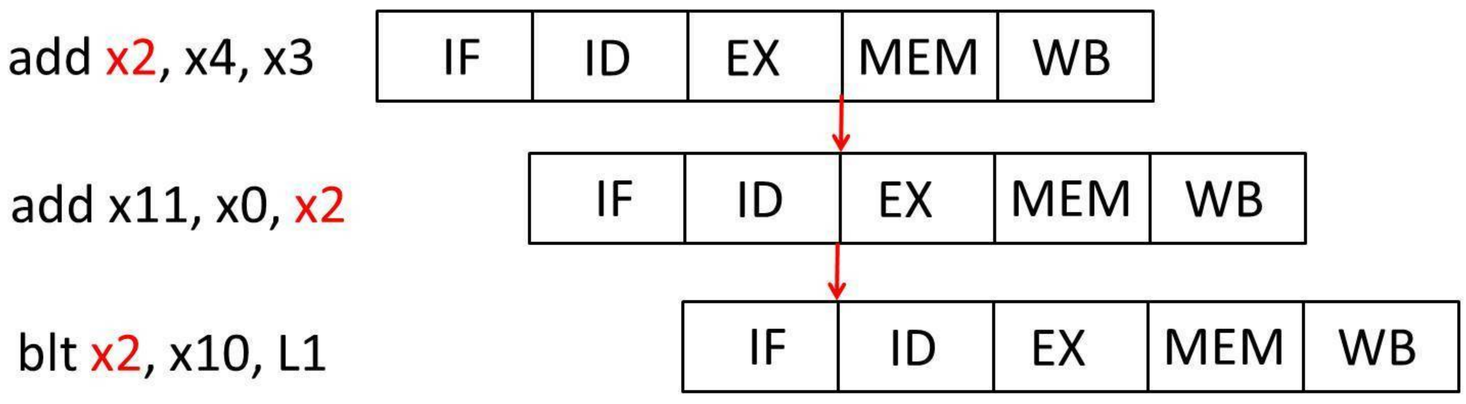
mul x8,x3,x5

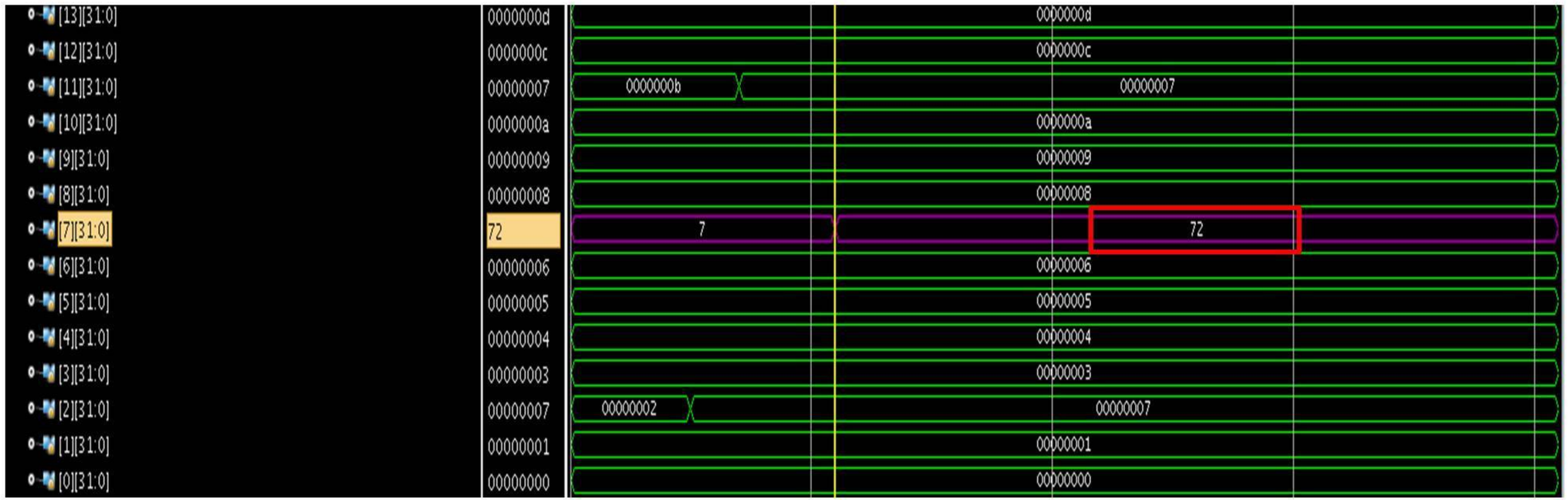


forwarding test 2

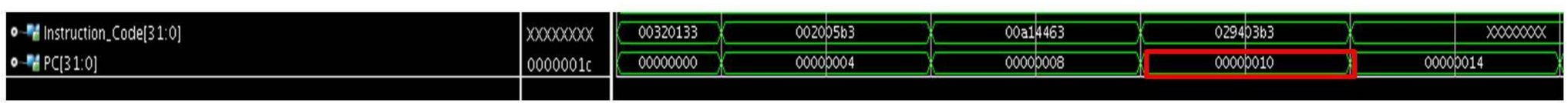
==========

```
add x2, x4, x3
add x11, x0, x2
blt x2, x10, L1
sub x7, x8, x9
L1:mul x7, x8, x9
```





The result has come from forwarded value



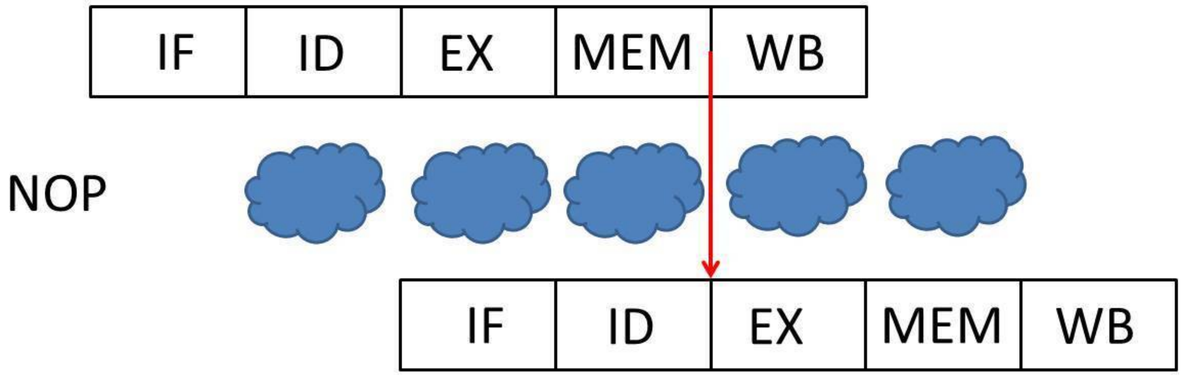
stall test

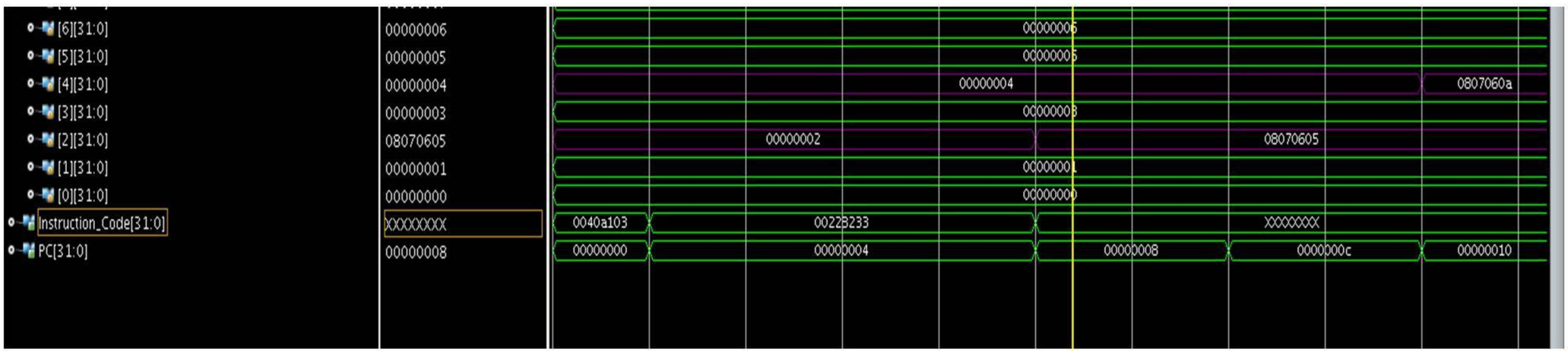
==========

lw x2,4(x1)

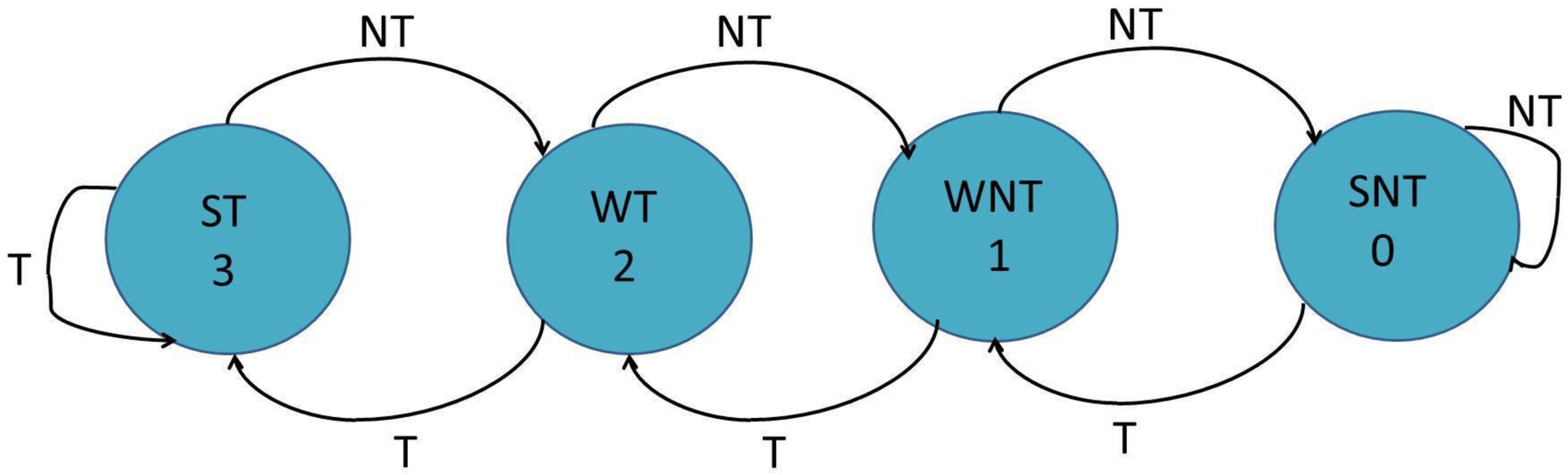
add x4,x5,x2

Stalling followed by forwarding





2 bit saturating counter prediction scheme



Branch equal to (beq) test

==========

beq x2,x2,L1

add x4,x5,x6

L1: add x7,x8,x9

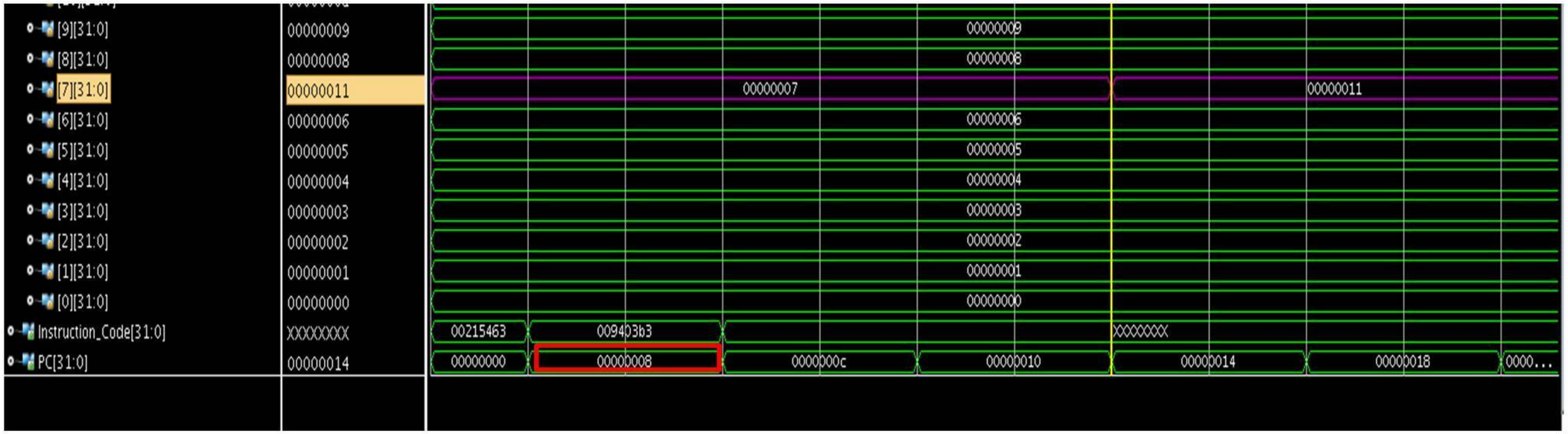
Branch greater than (bge) test

==========

bge x2,x2,L1

add x4,x5,x6

L1: add x7,x8,x9



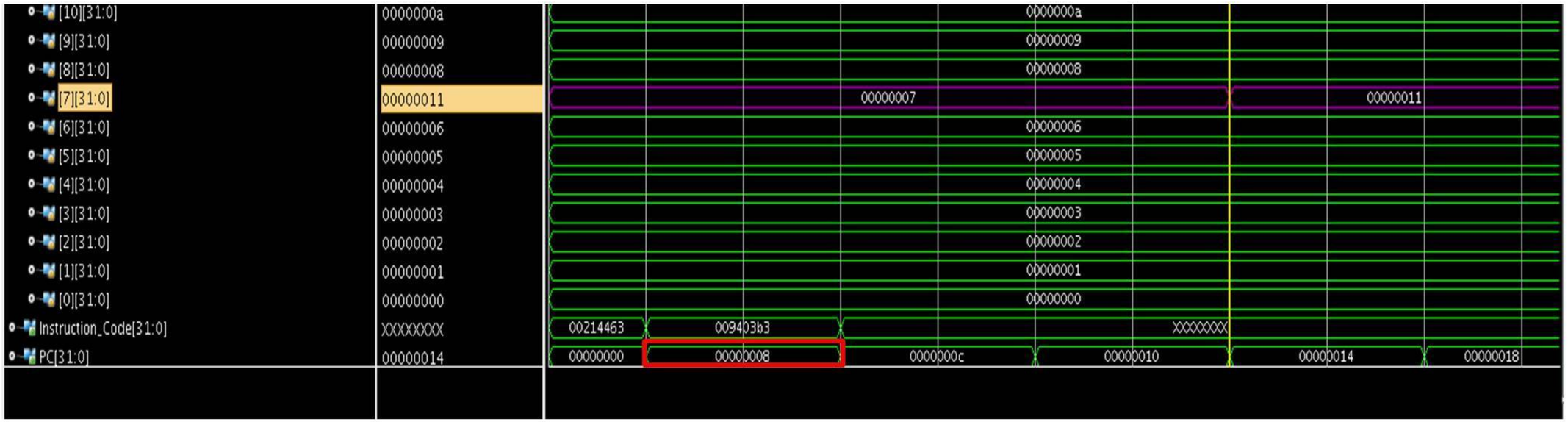
Branch less than (blt) test

==========

blt x2,x2,L1

add x4,x5,x6

L1: add x7,x8,x9



Application: Factorial

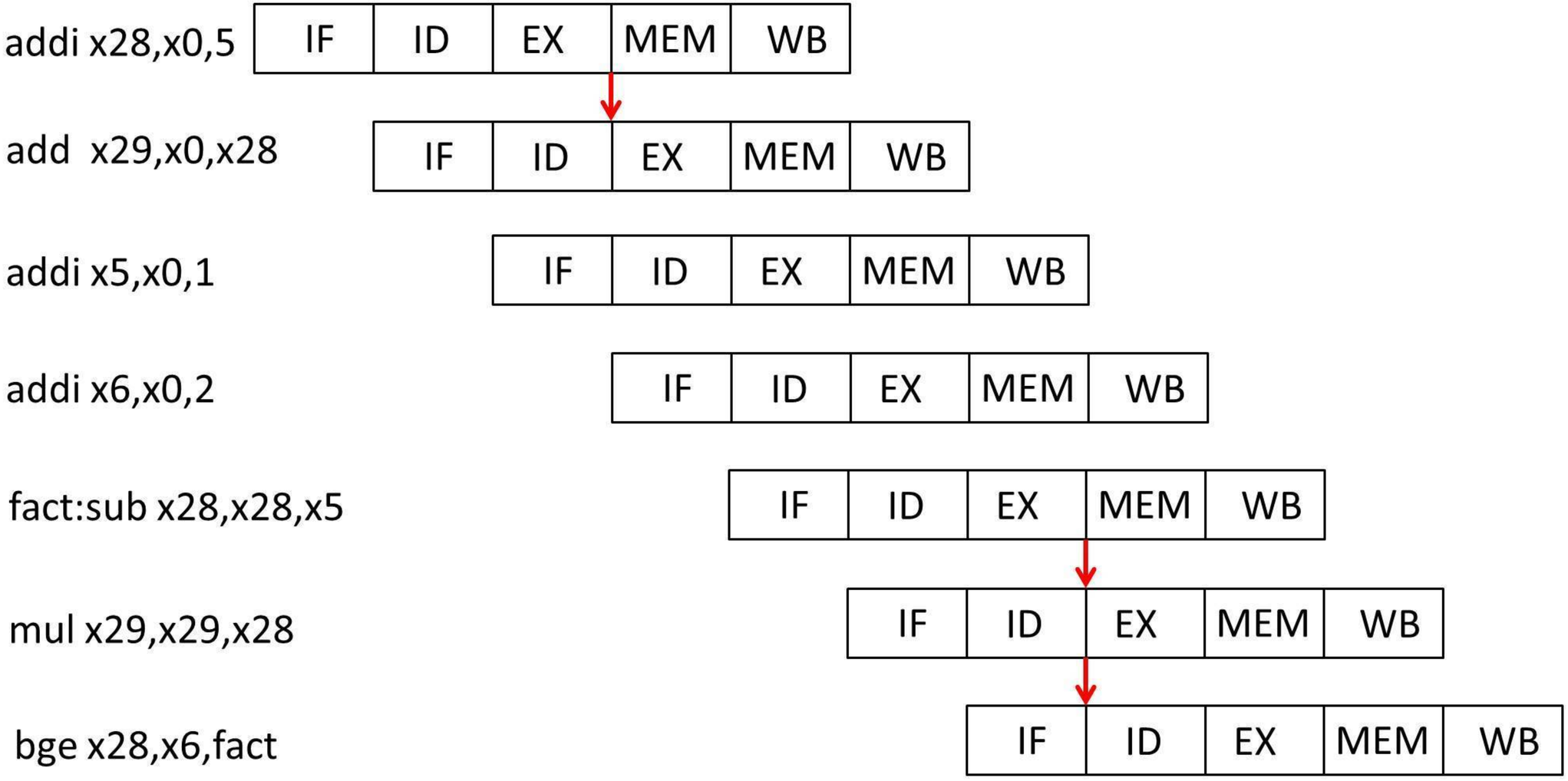
C program:

```
int main()

int a=5;                    //num=5
int x=a;
int temp1=1;
int temp2=2;

fact(int a)
{
if(a>temp2)
return n*fact(n-temp1);
}
```

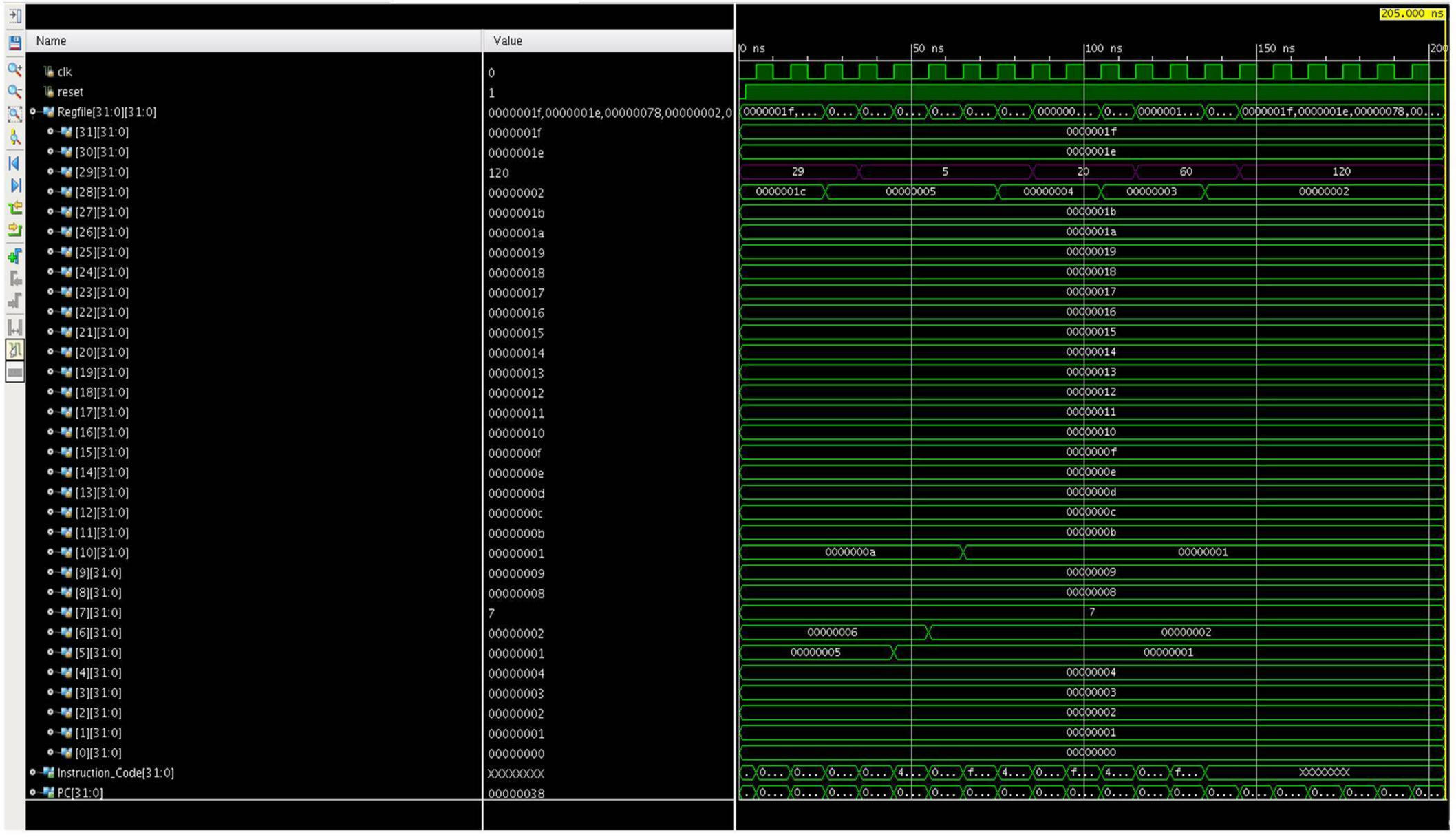
RISC V code for 5!

```
addi x28,x0,5                  //num = x28 = 5
add  x29,x0,x28
addi x5,x0,1
addi x6,x0,2
fact:sub x28,x28,x5
mul x29,x29,x28
bge x28,x6,fact
```

Result is stored in x29 register


addi x28,x0,5
IF ID EX MEM WB
add x29,x0,x28
IF ID EX MEM WB
addi x5,x0,1
IF ID EX MEM WB
addi x6,x0,2
IF ID EX MEM WB
fact:sub x28,x28,x5
IF ID EX MEM WB
mul x29,x29,x28
IF ID EX MEM WB
bge x28,x6,fact
IF ID EX MEM WB

Result


Name
Value
clk
0
reset
1
Regfile[31:0][31:0]
0000001f,0000001e,00000078,00000002,0
[31][31:0]
0000001f
[30][31:0]
0000001e
[29][31:0]
120
[28][31:0]
00000002
[27][31:0]
0000001b
[26][31:0]
0000001a
[25][31:0]
00000019
[24][31:0]
00000018
[23][31:0]
00000017
[22][31:0]
00000016
[21][31:0]
00000015
[20][31:0]
00000014
[19][31:0]
00000013
[18][31:0]
00000012
[17][31:0]
00000011
[16][31:0]
00000010
[15][31:0]
0000000f
[14][31:0]
0000000e
[13][31:0]
0000000d
[12][31:0]
0000000c
[11][31:0]
0000000b
[10][31:0]
00000001
[9][31:0]
00000009
[8][31:0]
00000008
[7][31:0]
7
[6][31:0]
00000002
[5][31:0]
00000001
[4][31:0]
00000004
[3][31:0]
00000003
[2][31:0]
00000002
[1][31:0]
00000001
[0][31:0]
00000000
Instruction_Code[31:0]
XXXXXXXX
PC[31:0]
00000038
205.000 ns
0 ns
50 ns
100 ns
150 ns
200